Workflow-based data parallel applications on the EGEE production grid infrastructure

Johan Montagnat, Tristan Glatard, Isabel Campos Plasencia, Francisco Castejon, Xavier Pennec, Giuliano Taffoni, Vladimir Voznesensky, Claudio Vuerli

Workflow-based data parallel applications on the EGEE production grid infrastructure

Johan Montagnat (johan@i3s.unice.fr)
CNRS, I3S laboratory, France. http://www.i3s.unice.fr/∼johan

Tristan Glatard (glatard@i3s.unice.fr)
CNRS, I3S laboratory - INRIA Sophia Antipolis, France.
http://www.i3s.unice.fr/∼glatard

Isabel Campos Plasencia (iscampos@unizar.es)
CSIC , Instituto de Fisica de Cantabria, Santander, Spain.
http://www.ifca.unican.es

Francisco Castejón (francisco.castejon@ciemat.es)
Laboratorio Nacional de Fusión. Asociación Euratom/Ciemat, 28040-Madrid, Spain. http://www-fusion.ciemat.es

Xavier Pennec (xavier.pennec@sophia.inria.fr)
INRIA Sophia Antipolis, France. http://www-sop.inria.fr

Giuliano Taffoni (taffoni@oats.inaf.it)
INAF-Osservatorio Astronomico di Trieste, Italy. http://www.ts.astro.it

Vladimir Voznesensky (vovic@nfi.kiae.ru)
Kurchatov Institute, Moscow, Russia. http://www.kiae.ru

Claudio Vuerli (vuerli@ts.astro.it)
INAF-Osservatorio Astronomico di Trieste, Italy. http://www.ts.astro.it

Abstract. Setting up and deploying complex applications on a grid infrastructure is still challenging and the programming models are rapidly evolving. Efficiently exploiting grid parallelism is often not straight forward. In this paper, we report on the techniques used for deploying applications on the EGEE production grid through four experiments coming from completely different scientific areas: nuclear fusion, astrophysics and medical imaging. These applications have in common the need for manipulating huge amounts of data and all are computationally intensive.

All the cases studied show that the deployment of data intensive applications require the development of more or less elaborated application-level workload management systems on top of the gLite middleware to efficiently exploit the EGEE grid resources. In particular, the adoption of high level workflow management systems eases the integration of large scale applications while exploiting grid parallelism transparently. Different approaches for scientific workflow management are discussed. The MOTEUR workflow manager strategy to efficiently deal with complex data flows is more particularly detailed. Without requiring specific application development, it leads to very significant speed-ups.

Keywords: workflows, workload management, data parallelism

1. Introduction

Scientific applications are often characterized by the large amount of data to be analyzed in order to produce relevant results. For the needs of scientific analysis, large data sets are processed through multiple data analysis codes, assembled in more or less elaborated *chains of processings* also known as *workflows* or *pipelines* among different communities. The EGEE grid infrastructure, composed of standard PCs interconnected through high performance links on the Internet, is providing a suitable infrastructure for handling the large number of computation tasks resulting from the execution of multiple-data workflows. In many cases when data segments can be processed independently, concurrent data processing is a massive, coarse grain parallelism that can efficiently be exploited to improve applications performances. The grid middleware is thus expected to provide a support for data parallelism but also a flexible workload management system addressing bulk jobs submission, group monitoring and workflows execution.

This paper focuses on the strategies explored for porting applications with different degrees of computation flow complexity on the EGEE grid infrastructure. The 4 applications presented are coming from 3 different scientific areas: nuclear fusion, astronomy and medical imaging. Although tackling very different problems, these applications exhibit common needs for intensive data processing. These applications properties and their performance on the EGEE grid infrastructure are described in section 2. The approaches adopted to address the application data parallelism needs and to control the computation flows on top of the EGEE workload management system are discussed in section 3. Section 4 discusses with more details the need for expressing and controlling complex application data flows. A review of well-known scientific workflow managers is made and the focus is put on the MOTEUR workflow manager designed to efficiently enact flows of application services.

2. Scientific Applications Description

The applications considered in this paper come from 3 different areas. First, two applications in the field of nuclear fusion relating to the simulation of plasma and confinement devices are presented. They are embarrassingly parallel kind of problems using the grid for its large scale data parallelism capability. The Planck experiment dedicated to cosmic background measurements is then introduced. It requires to process huge amounts of data in an application-specific workflow. The

last application presented is a medical image registration algorithm assessment method called *Bronze Standard*. It requires the execution of a complex application workflow and the manipulation of large input data sets. The grid parallelism is exploited through a generic workflow manager designed to efficiently handle such data-intensive applications.

2.1. Nuclear Fusion

Among the open theoretical problems in fusion plasmas, we are considering two different problems related to the confinement of fusion plasmas in specific devices called *stellarators*: kinetic transport of charged particles and stellarator optimization. These problems, in which a single particle or the microscopic scale is dominant, can be addressed by the kinetic theory using Monte Carlo techniques. This coarse grain parallelism makes efficient implementation on the EGEE grid infrastructure possible.

Kinetic Transport Simulation

Kinetic transport simulation aims at statistically estimating the properties of confinement from a large amount of charged particles considered [6]. The analysis of the results would proportionate an accurate description of what is happening in the stellarator as illustrated in Figure 1. A common approximation is to simulate individual particles without interaction between them, to introduce the electric and magnetic fields as effective fields, using as input the real measurements in the magnetic confinement device. The initial position in phase space of a large number of particles and the collisions with the background plasma can be estimated randomly.

We target the Spanish stellarator TJ-II [2], which is a medium size device of the "Heliac" type in operation in Madrid (Spain) since 1997. Its complex geometry makes difficult the development of numerical tools to study it. Since there are collective effects inside the plasma, producing physically meaningful results (as illustrated in Figure 1) require to consider a large number of particles. Our estimate is that an order of 10^7 particles need to be simulated. To give an order of magnitude, a run of 1000 particle trajectories on the TJ-II Stellarator produces about 1.5 GB of raw data. Currently, ions trajectories are simulated, but electrons should also be considered. Due to their mass (2000 times lower than ion ones), electron trajectories are more difficult to estimate and computations are expected to be 2000 times longer.

On the EGEE grid, the problem is tackled by sending similar jobs with different seeds for the Monte Carlo algorithm that distributes particles according to the given density and temperature. All jobs share common information (TJ-II geometry and background plasma

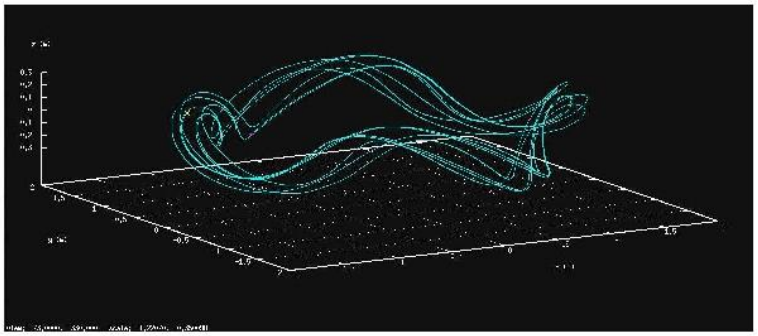

Figure 1. Particle trajectories in the TJ-II Stellarator

characteristics) stored in a fusion file catalog. This is a typical embarrassingly parallel kind of problem. The EGEE grid would permit to launch simultaneously a high enough number of interacting particles to produce relevant results.

Stellarators Optimization

Due to the large freedom in designing stellarators, it is necessary to optimize the 3D geometry of these devices [23] to determine the one with the best confining properties. The concrete output of an optimization process is a design in which neoclassical transport should be minimized, strategies for turbulence control should be available, and plasma equilibrium and stability should exist for high pressure plasmas. These requirements are sometimes contradictory and a trade-off must be reached by weighting them in the multi-criteria optimization process.

The plasma equilibrium in the stellarator can be found if the shape of the boundary plasma surface and the radial profiles of plasma pressure and toroidal current are prescribed. The boundary surface may be characterized by a set of Fourier harmonics that give the shape of the surface, the magnetic field, and the electric current. The TJ-II boundary is thus described by more than 150 harmonics. The resulting objective function to optimize takes about 20 minutes of computation time on a conventional PC.

Genetic algorithms have been chosen as optimization method. They consider the solution of the objective function as a variable "genome".

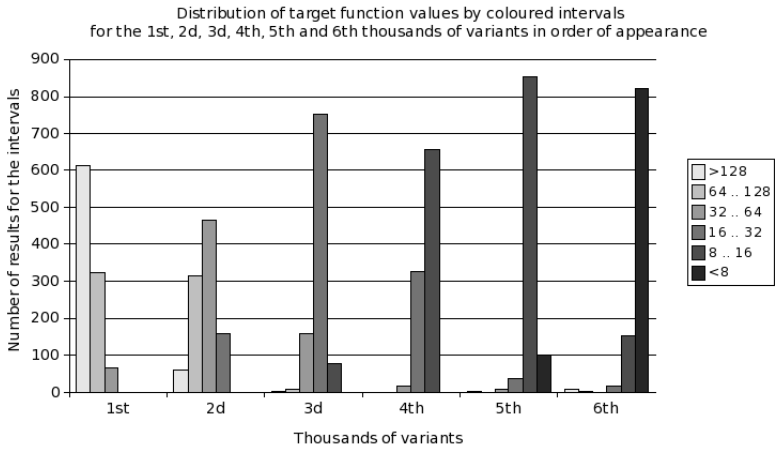

Figure 2. Genetic algorithm convergence.

At each iteration the algorithm proceeds with the selection of "parents", their breeding, and compute target function values for each "child" genome. Initial pools of genomes can be generated randomly inside the optimization parameters variation domain defined by the user. In the case of stellarators, two coefficients f and m for each one of the 150 Fourier harmonics from every parent vectors pair were bred separately. Every new coefficient is a random number of Gaussian distribution with mean $(f+m)/2$ and standard deviation $|f-m|/2$.

The genetic algorithms behave well for grid computations because genome pools can be processed asynchronously. Pools may be appended by grid job results sporadically, so aborting or delaying several jobs completion would not affect the overall optimization process badly. The parallelization resumes to a data parallel problem again.

A sample stellarator optimization task composed of 7.500 variant jobs has been executed on the EGEE infrastructure. About 1.500 of them were discarded since no equilibrium was found. The remaining 6.000 provided a parameter set solution. The overall computation took less than two days. Up to 70 variants were computed in parallel. The first, second, third, fourth and sixth thousand of the results in order of their return were aggregated into histograms (see Figure 2). The histograms represent the number of results falling into a given range of the target function value. The minimum of the target function used in the test is believed to have a value of several units. The sets of best values converge to the believed optimum value exponentially fast.

2.2. Planck Experiment: Cosmic Background Measurement

In recent years, large improvements have been carried out in Cosmic Background (CMB) detection. The ESA Planck satellite mission that will fly in 2007 [22, 28] aims at mapping the microwave sky with an unprecedented combination of sky and frequency coverage, accuracy, stability and sensitivity. Planck is composed of a number of microwave and sub-millimeter detectors grouped into a High Frequency (HFI) and a Low Frequency Instrument (LFI). Two Data Processing Centers (DPCs) are in charge of processing Planck data: one for HFI co-located in Paris (France) and Cambridge (UK), and one for LFI located in Trieste (Italy).

This space mission is extremely demanding in terms of storage and computational needs [3]. The LFI DPC has in charge the processing of approximately 100MB of compressed data each day for a total amount of approximately 100GB of raw data at the end of the mission. Starting from the raw data, the final product is a set of sky maps in the different wavelengths of the CMB and other important astrophysical data. One of the primary issues for the DPCs is to run a complete simulation of the Planck mission to test the data analysis pipeline. The simulation software must mimic the Planck observation procedure and any source of systematic effect. Moreover, it must cover all the aspects of the microwave sky [21].

The processing workflow, called *LevelS pipeline* [29], is depicted in figure 3. Only the main processing steps are shown: each step is composed by more than one pipeline stage sequentially linked and in total, 19 different algorithms are applied onto each input data set. Stages are chained by data dependencies (*e.g.* the output of stage A is used as input for stage B). At any given time, each detector observes the sky signal composed of a mixture of CMB (from the CMB power spectrum generated [34], a CMB map [16] is computed), galactic and extra-galactic foreground emissions convolved with the detector beam pattern. Instrumental noise is added according to detector characteristics. To simulate observations, a set of pointing directions for each mission simulation is generated according to the selected scanning strategy and satellite dynamical parameters. Finally, a Time Ordered Data (TOD) consisting of the time sequence of detector outputs is produced. The TOD is finally processed to extract the sky map of the CMB emission.

This workflow only represents the processings required to simulate one detector: for the LFI composed by 22 detectors (4 detectors at 30

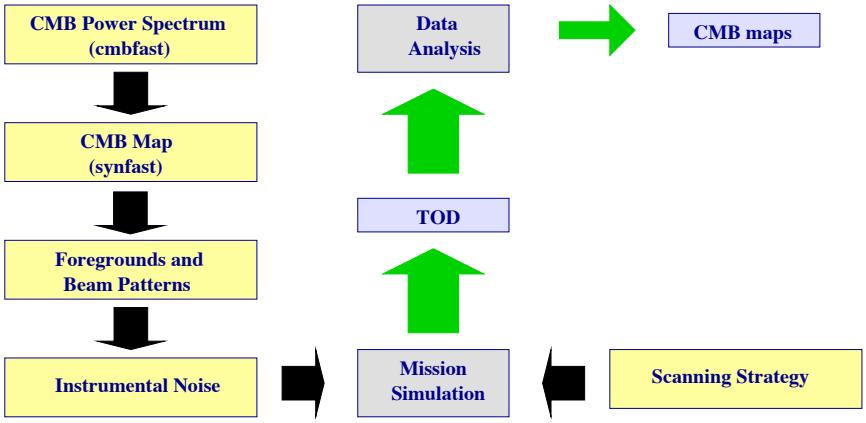

Figure 3. Level-S architecture.

GHz, 6 at 44 GHz and 12 at 70 GHz), 22 runs of this workflow with different input parameters are needed. In addition, the raw input data is made of circular samples acquired by the detectors during the rotation of the spacecraft around its spin axis. As a first approximation, we assume a perfect overlap between samples in two consecutive scan circles for a given pointing position. This reduces the number of sky signal simulation computations and results in a *short* run. However, the reality is more complex with no perfect overlaps (*long* run). The simulation time on a standard PC and the amount of data produced are reported in Table I: a sequential run takes more than 10 days (255.7 hours) to complete and produces 7 TB of uncompressed data. Furthermore, it is commonly necessary to explore the large space of parameters involved in each simulation (beam pattern, instrumental noise, foregrounds, the extragalactic point sources, the scanning strategies, etc).

A script-based workflow enactment procedure described in section 3.2 has been set up to control the execution of the PLANCK multi-data processing pipeline.

2.3. BRONZE STANDARD: MEDICAL IMAGE REGISTRATION ASSESSMENT

Medical image *registration* algorithms are playing a key role in a large number of medical image analysis procedures and therefore their accuracy is critical. Image registration consists in estimating the 3D transformation between two images, so that the first one can superimpose on the second one in a common 3D frame. A difficult problem, as for many other medical image analysis procedures, is the assessment of these algorithms robustness, accuracy and precision [17]. Indeed, there is no well established *gold standard* to compare to the algorithm results.

Table I. Computational time and data produced by the short and long LevelS simulations.

LFI 30GHz		LFI 44GHz		LFI 70GHz	
short	long	short	long	short	long
12min	389min	13min	623min	17min	834min
0.9GB	34.2GB	1.2GB	45.3GB	1.7GB	75GB
Total (4 30GHz + 6 44GHz + 12 70GHz)					
short		long			
5.5h 31GB		255.7h 1.3TB			

The *Bronze Standard* algorithm [11, 14, 25] is a statistical procedure that aims at estimating the accuracy of a given number of algorithms.

The idea is to compute the registration of a maximum number of image pairs with a maximum number of registration algorithms in order to obtain a largely overestimated system of transformation estimates (observations). From this redundant system, the *Bronze Standard* can be estimated by minimizing a specific criterion in the space of transformations to determine the transformations that better explains the observations. The accuracy of a given algorithm is then computed as the distance between its results and the Bronze Standard. The higher the number of independent registration algorithms considered and the number of images processed, the more accurate the procedure. It makes this application very data-intensive.

The Bronze Standard procedure exhibits the complex workflow illustrated in Figure 4. On this figure, each box represents an application service to be executed for every image in the tested database. In addition to registration algorithms themselves, additional processing stages are needed for pre-processing the images, initializing the computations, making format conversions and quantitatively analyzing the registration results. Box colors represent the current execution status in the graphical user interface for a workflow being executed. The arrows between services represent either data dependencies (the output of a service is piped into the input of the following one) or temporal dependencies (synchronization barriers) as detailed in section 4.2. In addition to the computational services, the diagram represents data sources (two triangles representing two input image sets to be registered) and sinks (output collectors, diamonds). The graph represents the flow of

processings, described in the Scufl XML-based language [26]. The data to process is described independently. To efficiently exploit the EGEE grid parallel capability, we implemented the MOTEUR workflow enactor[1] [9, 10], an optimized manager for Scufl workflows. From the application graph of processings and given input data sets, the engine dynamically determines the data flows to be processed. Using the rich semantics of the Scufl data composition operators discussed in section 4.3 [24, 26], this results in the production of a very large number of computation tasks, many of which can be executed in parallel although some dependencies have to be taken into account in the scheduling.

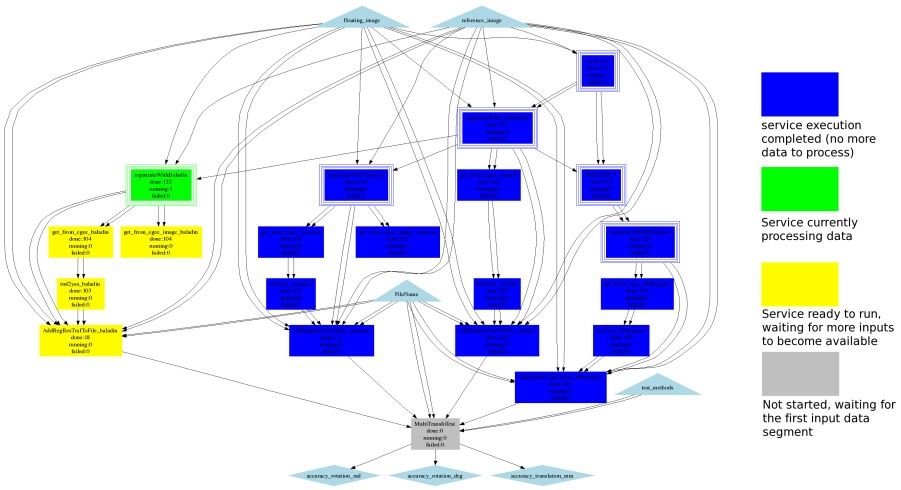

Figure 4. Bronze Standard application workflow: each box represents an image processing service and arrows show dependencies.

In a typical run, the computation flow has to be reiterated over hundreds of image pairs. This causes the submission of thousands of computation tasks to the grid. The sequential computation time of a typical run that would be in the order of 3 days on a regular PC is thus reduced to approximately 3 hours on the EGEE grid.

3. Computation Flows Description and Management

All applications presented exhibit a large data parallelism and various degrees of computation flow complexity. Depending on the application characteristics, different strategies have been adopted for porting it on

[1] MOTEUR workflow enactor, `http://egee1.unice.fr/MOTEUR`

the EGEE grid infrastructure. This section describes the application layers developed to deal with the data parallel jobs.

3.1. Fusion applications: Data Parallel Jobs

Both fusion applications (kinetic transport simulation and stellarator optimization) are typical examples of embarrassingly parallel problems: a very large number of jobs involving the same processing but on independent data sets can be triggered.

In the case of the kinetic transport problem, hundreds of particle trajectory simulation jobs can be executed concurrently. On each run, a different initialization random seed indicates essentially the starting position and momentum of the particles in the magnetic confinement device, but the computations remain identical. A strategy to submit, control and retrieve the results from hundreds of jobs that do not require tedious user intervention is needed.

Similarly, genetic algorithms used for stellarator optimization are producing identical computations on different input "genes". The model implemented in this case is slightly more complex since all genes cannot be generated at once at the beginning of the execution. Starting from a pool of initial genes, new genes have to be generated during the execution that will be used for initializing new computing tasks.

A set of Python scripts as been set up for this purpose. One of them generates an initial genome pool, another one spawns new jobs, the third gathers already computed results from the grid and the fourth generates new part of genome pools from the existing ones. The number of concurrently spawned jobs is kept below a given threshold. The iteration is realized by a master *bash* script which loops over the different steps.

The current EGEE production middleware describes computation tasks through a CONDOR ClassAds-based Job Description Language (JDL). Each JDL file specifies both the binary to execute (the processing to perform) and the input/output files (the data to process) either directly (explicit mention of the input and output data files) or indirectly (through the job input sandbox, command line parameters etc). This job description technique is shared with many middlewares exploiting a *global computing* approach such as GLOBUS[2], CONDOR[3] or LCG/gLite[4]. As a consequence, hundreds of similar job description files are generated by the application layer. They only differ in the input data (different seeds in this case), but not in the kind of processings

[2] GLOBUS Toolkit, `http://www.globus.org`
[3] CONDOR, `http://www.cs.wisc.edu/condor`
[4] gLite middleware, `http://www.glite.org`

described. Alternatively, a different approach is exploited by applications structured as architectures of services. This *metacomputing* model is implemented in middlewares which invoke application codes instrumented with a standard interface. For instance, the DIET [5], Ninf [30] or Netsolve [1] middlewares use the GridRPC remote invocation standard. The service Interface Description Language (IDL) enables the description of the service invocation method (processing) independently from the data to process. Data is delivered by the application dynamically, at execution time. An intermediate task description file is thus not needed for each processing with the same pattern.

In a near future, the gLite v3.1 middleware is announced to support a new *parametric* kind of job in the JDL. A parametric JDL is a job template where a specific parametric field (*e.g.* a counter numbering input data segments) can be used inside the JDL. The parameter is variable and will take any value in a user specified range with a given increment. At job execution time, the job template is instantiated into as many jobs as the parameter value can take and a group of jobs with similar processings but differing in their parameters will be submitted. Although less flexible than the metacomputing model (in particular, only a single parameter is accepted), this strategy will significantly tighten the gap between both approaches from an application point of view. Besides, it enables performance optimization through bulk submission of all jobs. It will lighten the application code by handling more complex job descriptions at the middleware level.

In the case of kinetic transport application, parametric jobs can directly be used to describe computations needed. For stellarators optimization, all jobs cannot be described at once but gene pools can be described in a single parametric job.

3.2. Planck Simulations: Script-based Pipeline

A single Planck run involves 22 detector simulations. The application was ported to the EGEE grid using an interactive configuration and submission program that produces JDLs automatically. Each JDL describes a complete pipeline: the workflow structure is embedded inside an execution script that is sequentially executed on an EGEE Worker Node (WN). The configuration program can produce a single job that chains all the 22 simulations on a single WN, a swarm of 22 independent jobs (one for each detector) to be executed on different WNs, or any intermediate degree of parallelization. It creates all the input configuration files needed by the pipeline stages and the pipeline execution script accordingly. It also tunes the execution for the set of cosmological

and instrumental parameters, the set of radiometers and the kind of detectors requested by the user.

The application script was designed to be modular and to ease the integration of new pipeline stages when some upgrade is needed. The performance of the application was measured by running 20 simulation runs and averaging the total execution time (including submission time, execution time and data saving time). With maximum parallelization (22), the grid average execution time is 1355 minutes for long runs (30 minutes for short runs respectively). Compared to the 255.7 hours (resp. 5.5 hours) workstation execution time, this leads to a speed-up of 11 in both cases. This number is only half of the ideal speed-up that could be expected in this case (22 independent data parallel jobs) but this is not surprising: the EGEE grid is under permanent load and job submissions are delayed by variable batch queuing times and middleware overhead. The measures were averaged over 20 different runs to reduce the effect of variable load conditions on the infrastructure. They thus give a realistic speed-up in production conditions. A much higher degree of parallelism can be expected as many independent Planck runs will be needed for exploring the ranges of parameters and tuning the pipeline.

By embedding the pipeline execution script into a single job, it is not possible to process the different stages of the pipeline independently, possibly concurrently. This reduces the level of parallelism that can potentially be achieved. However, this also drastically reduces the grid payoff in the execution. Indeed, running independent stages would lead to independent job submissions, each of them being queued and delayed before their execution. Given that the LevelS pipeline is sequential, these delays accumulate before the completion of a single detector simulation. On the EGEE production infrastructure, the payoff observed for each job submission in commonly in the order of 5 minutes. A single detector simulation takes in the order of 15 minutes for short runs on a workstation and 11.6 hours for long runs. Clearly, splitting a short run in independent stages would introduce a total payoff in the order of 100 minutes (19 sequential stages times the payoff) far too high compared to the application execution time. For a long run, the payoff is proportionally lower, although not negligible. In addition, chaining the stages of the pipeline on a single WN saves data transfers. Indeed, all data outputs of a given stage can be stored into local files and read as inputs of the following stage directly. Splitting the execution of the pipeline would lead to penalizing large data transfers between each stage and a data transfer manager would be needed in the workflow management system.

It should be noted that the above discussion on parametric jobs holds for the Planck pipeline as well: the gLite 3.1 parametric job de-

scription system would help in generating and executing the 22 similar pipelines composing a single data parallel run.

3.3. Bronze Standard: Data Intensive Workflow Engine

We adopted a service-based approach for the Bronze Standard application as it provides a high degree of flexibility and standard code invocation procedures. The MOTEUR workflow engine invokes Web-Services containing the application code. The Scufl description language defines a link to a WSDL document for each service node in the workflow graph. In theory, any Web-Service can thus be enacted. However, only specifically instrumented Web-Services will trigger an execution on the EGEE grid infrastructure. In a classical Service-Oriented Architecture, services are black boxes and there is no information on the way the service is executed. In the case of the Bronze Standard application, we are interested in a remote execution of all services to the grid. Since application services cannot directly be deployed on the EGEE grid infrastructure, we use a specific submission service that interfaces Web-Service invocations with the EGEE workload management system. Furthermore, all application codes reused to compose the Bronze Standard workflow are standard executables that were not specifically instrumented with a Web-Service interface. The submission service acts as a Web-Service wrapper for legacy codes. It does not require any modification nor recompilation of application codes. Only a small XML document describing the code invocation command line is requested from the developer [8].

Similarly to the other applications reported in this manuscript, data parallelism plays a predominant role in the application performance optimization. However, the Bronze Standard workflow also exhibits two other levels of code parallelism. The first one is the workflow graph parallelism that can be seen on the graph displayed in figure 4: two branches of the graph with no dependencies can be executed in parallel. In addition, the multi-data nature of this representation enables services parallelism, also known as *pipelining*: two sequential services can process two different data segments independently. The MOTEUR workflow manager is optimized to transparently exploit these three levels of parallelism (data, workflow and service parallelism) [9]. Given the workflow description graph defined in Scufl and given input data sets, MOTEUR services dynamically generate jobs to be executed on the EGEE infrastructure. Each job is described through a classical JDL corresponding to the execution of a given image processing algorithm on a given input data segment. The services use the EGEE workload management interface to submit and monitor it. As many parallel jobs

as possible are submitted. On completion of one computing task, the resulting data is piped into following services and new tasks are created. The process finishes when no more data is produced by any service in the workflow.

Note that each job submitted by MOTEUR is impacted by the grid payoff as discussed in the case of the PLANCK experiment. The workflow engine maximizes the parallelism achievable but as a consequence, the number of jobs increases and the latencies for each of them accumulate in the sequential branches of the workflow. In order to optimize the application execution time, a non trivial trade-off has to be found between the granularity of the tasks submitted. Earlier attempts have been made to optimize the data granularity of data parallel tasks on a grid infrastructure, either considering experimental data [32] or modeling the grid infrastructure through a probabilistic approach [12]. Adapting these methods to workflows is made complex due to dependencies between tasks and new execution models are being considered [13]. In MOTEUR, a heuristic based on a jobs grouping strategy has been implemented to lower this grid payoff [8]. The idea is that in the case of sequential branches in the workflow, one always gain to group several stages in a single job (PLANCK strategy) as they will be executed sequentially anyway, but the lesser the number of jobs, the lower the latency paid and the lower the intermediate data transfers needed. Grouping two services in a single one for the execution on a grid infrastructure is not a straight forward problem. In the case of real black-box services, it is generally not possible to combine the code of the services in a single execution chunk. This is only feasible in our case thanks to the use of the generic submission Web-Service: the specific knowledge on this service enables the codes sequential grouping.

Figure 5 illustrates the benefits of MOTEUR optimization strategies on the Bronze Standard application. The abscissa represents the scale of the problem (the number of pairs of input images to process). The ordinate gives the corresponding execution time (in hours). The highest `NOP` (No OPtimization) curve represents the time needed for a naive grid execution. Naive execution means that only workflow parallelism is exploited (this correspond to the baseline functionality available in every workflow managers). The execution requires more than 35 hours to process a 126 image pairs data set. The `JG` curve correspond to the gain obtained by job grouping: the latency reduction saves a couple of hours on the complete workflow. The `DP` curve corresponds to the data parallelism. This is the highest performance improvement as expected. The slope of the `DP` curve being smaller than the slope of the `NOP` curve, the speed-up increases with the number of data segments. Finally, the

`SP` (service parallelism) and `JG` optimizations are added to provide the lowest curve. The total execution time falls down to less than 3 hours.

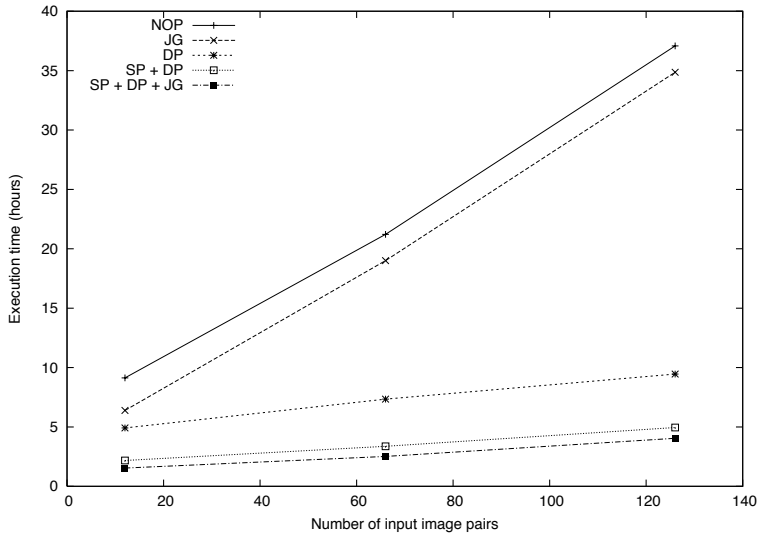

Figure 5. Bronze Standard application performance improvement using MOTEUR optimization strategies

MOTEUR is a generic workflow manager not bound to the Bronze Standard specific application. This extra layer between the application and the EGEE middleware fully hides the middleware to the application developer by taking care of jobs submission and middleware feedback interpretation. It transparently enables parallel execution of the application. It thus provides a flexible and evolutive framework for efficiently porting a data-intensive application workflow on the EGEE grid. MOTEUR has also been interfaced to DIET GridRPC services and the OAR batch scheduler [4].

4. Scientific workflows

4.1. Application Workload Management

The applications presented in this paper are all characterized by the need for processing large and independent data sets using a single computation flow. They exhibit different levels of workflow description and management needs:

- In the simplest case (kinetic transport) the application is a pure embarrassingly parallel problem. The tools needed are concerning the submission, monitoring, and data collection for hundreds to thousands of concurrent jobs. Most of this work is still handled at the application level today. The EGEE grid proposes tools for grouping jobs submission, monitoring and fetching data but little is done for error recovery and execution completion in case of partial failure. The parametric job type introduced in gLite 3.1 should facilitate the development process of this kind of application in the future.

- When several computation steps are needed for data preparation, processing and analysis (stellarator optimization, Planck experiment), scripting is an easy way of prototyping and enabling the application on the grid infrastructure although it is very application-specific (static description of an application workflow). It requires significant effort for each new application. Furthermore it only provides the limited quality of service that could be implemented at this level. Parallelization of an application script may be non trivial.

- Finally, the execution of complex, data parallel workflows which can benefit from other levels of parallelism (Bronze Standard) are better handled by a generic workflow engine. It provides a flexible and extensible framework for describing an application flow and controlling its execution. The workflow manager decouples the application description from its grid execution. It provides transparent parallelization of the application and hides the middleware technical details. High level workflow enactors are therefore expected to provide a generic solution for easing multiple scientific applications development and grid deployment.

Workflow managers are the subject of extensive studies inside and outside the grid community. The next session discusses different approaches and identifies the key points concerning scientific applications according to our experience.

4.2. Scientific workflows description

Workflow managers have been developed in very different contexts. Seminal workflow managers were designed for describing complex business processes. The flagship business workflow description language is BPEL [19] which can be enacted through a large armada of workflow engines. Such managers are characterized by a strong focus on the *control* flow. In fact, they are complete programming languages with typed variables, expressions, conditional and loop control structures. With a graphical design interface they offer a high level programming environment for non specialists. There is *a priori* no parallelism possible with these languages, although a *for-all* kind of loop is planed in the next BPEL standard release. The JOpera tool [27] is another recent example of a business workflow manager supporting complex parallel structures.

In the scientific area, more data-centric managers have been proposed. They usually provide a more limited panel of control structures as they rather focus on the execution of heavy-weight algorithms designed to process large amounts of data. The complex application logic is supposed to be embedded inside the basic application components. The scientific workflow description languages are not so rich but the execution engines are better taking into account execution efficiency and data transfer requirements.

A detailed taxonomy of scientific workflow systems is proposed by Yu and Buyya [33]. The authors characterize the workflows managers according to their design method, scheduling strategy, fault tolerance and data transfer capabilities. Despite this detailed classification, they do not distinguish workflow managers exploiting the global computing model versus the metacomputing model.

The emblematic global computing workflow manager is the Directed Acyclic Graph Manager (DAGMan[5]) that is available in the latest version of the gLite middleware. This system basically allows the description of precedence constraints between Condor jobs. It is the basis for several higher level layers dealing with optimization of tasks submission such as the Pegasus [7] system. A DAGMan workflow is composed of computing tasks and temporal dependencies between tasks. There is no notion of data flows in DAGMan: the output of a job A happens to be reused by a job B only if A and B's JDL are written so that there respective output and input data files match. There is no data transfer handled by the manager itself. As a secondary consequence, there are no possible loops in DAGs (hence the name: Directed *Acyclic* Graph). A loop would create an endless chain of temporal dependencies.

[5] Condor DAGMan, `http://www.cs.wisc.edu/condor/dagman/`

The metacomputing approach applied to workflows provides more flexibility in the sense that the graph of dependencies represents either data dependencies between processes or control dependencies. Control links impose temporal dependencies between workflow processes. They can be used to implement data synchronization barriers which might become necessary for some applications due to the parallel execution of multiple data flows. There may not be loops on control links just as with DAGs but nothing prevents loops on data links. Furthermore, all services are not necessarily invoked: only branches receiving data are executed. Therefore it is straight forward to describe conditional and looping control structures. The data parallelism is implicit in this model: data is described independently from the computational services. Therefore, two different invocations with two different data sets do not require any modification of the computational graph but they lead to different data flows. Data transfers can be handled at the workflow management level since data sets are known from the workflow enactor. Well known examples of scientific workflow managers exploiting the metacomputing model are the Kepler system [20], the Taverna workbench [26] and the Triana workflow manager [31]. However, they are not natively interfaced to a grid infrastructure and they require adaptations to exploit parallel grid resources. The P-GRADE portal [18] workflow manager is a grid-enabled workflow manager which aims at exploiting both approaches: originally interfaced to DAGMan, it has recently been extended to support so called *parametric studies* [15]. A parametric workflow is a template-based workflow for which multiple tasks will be instantiated similarly to the way the gLite parametric jobs produce multiple task executions. P-GRADE parametric studies are more flexible though, as multi-dimensional parameter spaces can be explored and complex data flows can be described. Parametric workflows are enacted either through MOTEUR or through a DAG generator which automatically instantiates all tasks corresponding to a given parametric workflow once the input data set is known.

4.3. Data flows description

Data flows description plays a major role in scientific workflows, as illustrated by the four applications introduced in this paper, for several reasons. The problems addressed are massively parallel and the maximum performance improvement can be expected from data parallel execution. Furthermore, the data location and transfer strategies have a significant impact on the application performances: the workflow graph topology is not a sufficient input to minimize execution time.

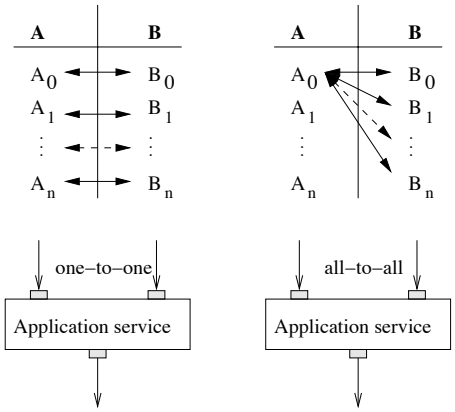

Figure 6. Action of the *one-to-one* (left) and *all-to-all* (right) operators on the input data sets

By decoupling data and computation flows, the metacomputing approach enables the definition of very interesting data composition strategies. In the metacomputing model, a service receives data segments on all its inputs periodically (and asynchronously in case of parallel execution) for processing. Data composition patterns are used to describe how different data inputs are combined for processing. There are two basic data composition patterns, very frequently encountered in scientific applications, that were first introduced in the Scufl language. They are illustrated in figure 6. Let $\mathbf{A} = \{\mathrm{A}_0, \mathrm{A}_1, \ldots, \mathrm{A}_n\}$ and $\mathbf{B} = \{\mathrm{B}_0, \mathrm{B}_1, \ldots, \mathrm{B}_m\}$ be two input data sets.

The *one-to-one* composition pattern (left of figure 6) is the most common. It consists in processing two input data sets pairwise in their order of arrival. This is the classical case where an algorithm needs to process every pair of input data independently. An example is a matrix addition operator: the sum of each pair of input matrices is computed and returned as a result. Usually, the two input data sets have the same size ($m = n$) when using the one-to-one operator, and the cardinality of the results set is $m = n$. If $m \neq n$, a semantics has to be defined (*e.g.* process the $\min(m, n)$ first pairs).

The *all-to-all* composition pattern (right of figure 6) corresponds to the case where all inputs in one data set need to be processed with all inputs in the other data set. A common example is the case where all pieces of data in the first input set have to be processed with all parameter configurations defined in the second input set. The cardinality of the processed data set is $m \times n$.

Combining the two operators introduced above enables very complex data composition patterns. The pairwise one-to-one and all-to-all

operators can be combined to compose data patterns for services with an arbitrary number of input ports. In this case, the priority of these operators needs to be explicitly provided by the user. By assigning data composition operators to all pairs of inputs in the workflow services, users can describe non trivial data flows whose exact topology depends on the variable workflow input data sets. Composition operators are a powerful tool for data-intensive application developers who can represent complex data flows in a very compact format. Although the one-to-one operator preserves the input data sets cardinality, the all-to-all operator may lead to drastic increases in the number of data to be processed. It may lead to the production of enormous amounts of tasks despite the apparently simple topology of the application workflow graph. The Bronze Standard application make intensive use of the data composition patterns to describe its data flow. The fully deployed graph of the application, a thousands node graph, would not be displayable. Thanks to the compact representation provided, it resumes to the rather simple graph displayed in figure 4.

Handling the data composition strategies in a service and data parallel workflow is not straightforward. Indeed, the data composition results for a given service cannot be computed once for all the data sets. Indeed, due to service parallelism, the input data segments of a service are received one by one. The data composition thus has to be recomputed each time a new data segment is received and the service has to record the data sets it has already processed. Moreover, data provenance has to be properly tracked in order to compute one-to-one composition operators. Indeed, due to data parallelism, a piece of data is able to overtake another one during the processing and this disordering could lead to a causality problem. Besides, due to service parallelism, several data sets are processed concurrently and one cannot number all the produced data once computations completed. MOTEUR implements a data provenance strategy to sort out the causality problems that may occur. Attached to each processed data segment is a history tree containing all the intermediate results computed to process it. This tree unambiguously identifies the data. This data provenance information is also often of high interest in a scientific data analysis context.

5. Conclusions

This paper detailed the design and deployment of four scientific applications from different scientific areas with a common need for large data sets analysis. Grids are naturally very well suited for processing data intensive parallel applications (embarrassingly, coarse-grain

parallel problems). The EGEE grid middleware provides a foundation layer for implementing such applications but it still delegates to the application the handling of complete data sets. Higher level workload management is expected to further ease data-intensive applications development. Recent development in the gLite workload management system show that these requirements are progressively taken into account and integrated at the level of the middleware.

Workflow managers provide a generic way of dealing with scientific application requirements: they enable transparent parallelization of the application execution, they provide an interface layer hiding the grid complexity to the application developer and they can address complex data flows description and management problems. There exists a wide variety of workflow managers adopting different approaches, from extensions of traditional batch systems (graphs of tasks) to service-based composition. All workflow representation frameworks do not provide the same flexibility nor expressiveness and application developers should carefully look at their application data flow needs and the kind of application design they are targeting.

We are developing the MOTEUR workflow enactor. Based on the Scufl scientific workflow description language, it targets high performance while taking advantage of the flexibility provided by data composition operators to describe complex data flows in a compact format.

Acknowledgments

The work on the MOTEUR workflow manager is partially funded by the AGIR and the GWENDIA projects (contract number ANR-06-MDCA-009) from the French National Research Agency MDCA program. We are grateful to the EGEE European project (contract number INFSO-RI-508833) for providing resources and support to the experiments described in this paper.

References

1. Arnold, D., S. Agrawal, S. Blackford, J. Dongarra, M. Miller, K. Seymour, K. Sagi, Z. Shi, and S. Vadhiyar: 2002, 'Users' Guide to NetSolve V1.4.1'. Technical Report ICL-UT-02-05, University of Tennessee, Knoxville.
2. Ascasíbar E. *et al*: 2002, 'Confinement and stability on the TJ-II Stellarator'. *Plasma Physics and Controlled Fusion* **44**, B307.
3. Bond, R., R. Crittenden, A. Jaffe, and L. Knox: 1999, 'Computing Challenges of the Cosmic Microwave Background'. *Computers in Science & Engineering* **1**(1), 21–29.

4. Capit, N., G. Da Costa, Y. Georgiou, G. Huard, and C. Marti: 2005, 'A batch scheduler with high level components'. In: *Cluster computing and Grid 2005 (CCGrid'05)*, Vol. 2. pp. 776– 783.
5. Caron, E. and F. Desprez: 2005, 'DIET: A Scalable Toolbox to Build Network Enabled Servers on the Grid'. *International Journal of High Performance Computing Applications*.
6. Castejón, F. *et al*: 2006, 'Ion orbits and ion confinement studies on ECRH plasmas in TJ-II stellarator'. *Fusion Science and Technology*.
7. Deelman, E., J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda: 2003, 'Mapping Abstract Complex Workflows onto Grid Environments'. *Journal of Grid Computing (JGC)* **1**(1), 9–23.
8. Glatard, T., D. Emsellem, and J. Montagnat: 2006a, 'Generic web service wrapper for efficient embedding of legacy codes in service-based workflows'. In: *Grid-Enabling Legacy Applications and Supporting End Users Workshop (GELA'06)*. Paris, France.
9. Glatard, T., J. Montagnat, D. Lingrand, and X. Pennec: 2007a, 'Flexible and efficient workflow deployement of data-intensive applications on grids with MOTEUR'. *International Journal of High Performance Computing and Applications (IJHPCA)*.
10. Glatard, T., J. Montagnat, and X. Pennec: 2006b, 'Efficient services composition for grid-enabled data-intensive applications'. In: *IEEE International Symposium on High Performance Distributed Computing (HPDC'06)*. Paris, France.
11. Glatard, T., J. Montagnat, and X. Pennec: 2006c, 'Medical image registration algorithms assesment: Bronze Standard application enactment on grids using the MOTEUR workflow engine.'. In: *HealthGrid conference (HealthGrid'06)*. Valencia, Spain.
12. Glatard, T., J. Montagnat, and X. Pennec: 2006d, 'Probabilistic and dynamic optimization of job partitioning on a grid infrastructure'. In: *14th euromicro conference on Parallel, Distributed and network-based Processing (PDP06)*. Montbéliard-Sochaux, France, pp. 231–238.
13. Glatard, T., J. Montagnat, and X. Pennec: 2007b, 'Optimizing jobs timeouts on clusters and production grids'. In: *International Symposium on Cluster Computing and the Grid (CCGrid)*. Rio de Janeiro.
14. Glatard, T., X. Pennec, and J. Montagnat: 2006e, 'Performance evaluation of grid-enabled registration algorithms using bronze-standards'. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI'06)*. Copenhagen, Denmark.
15. Glatard, T., G. Sipos, J. Montagnat, Z. Farkas, and P. Kacsuk: 2007c, *Workflow Level Parametric Study Support by MOTEUR and the P-GRADE Portal*, Chapt. 18. Springer.
16. Gorski, K. M. *et al*: 1998, 'Analysis issues for large CMB data sets'. In: *Evolution of large scale structure : from recombination to Garching*. Garching, Germany : ESO, p. 37.
17. Jannin, P., J. Fitzpatrick, D. Hawkes, X. Pennec, R. Shahidi, and M. Vannier: 2002, 'Validation of Medical Image Processing in Image-guided Therapy'. *IEEE Transactions on Medical Imaging (TMI)* **21**(12), 1445–1449.
18. Kacsuk, P. and G. Sipos: 2005, 'Multi-Grid, Multi-User Workflows in the P-GRADE Grid Portal'. *Journal of Grid Computing (JGC)* **3**(3-4), 221 – 238.

19. Khalaf, R., N. Mukhi, and S. Weerawarana: 2003, 'Service-Oriented Composition in BPEL4WS'. In: *International World Wide Web Conference (WWW)*. Budapest, Hungary.
20. Ludäscher, B., I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. Lee, J. Tao, and Y. Zhao: 2005, 'Scientific Workflow Management and the Kepler System'. *Concurrency and Computation: Practice & Experience*.
21. Maino, D., C. Burigana, and T. Maltoni: 1999, 'The Planck-LFI instrument: Analysis of the 1/f noise and implications for the scanning strategy'. *Astronomy & Astrophysics (A&A)* **140**(1), 383–392.
22. Mandolesi, N., C. Lawrence, F. Pasian, M. Bersanelli, C. Butler, and et al: 1998, 'Planck LFI'. *Proposal submitted to ESA* **1**(1), 1–140.
23. Mikhailov, M., V. Shafranov, A. Subbotin, and etal.: 2002, 'Improved alpha-particle confinement in stellarators with poloidally closed contours of the magnetic field strength'. *Nuclear Fusion* **42**, L23–L26.
24. Montagnat, J., T. Glatard, and D. Lingrand: 2006, 'Data composition patterns in service-based workflows'. In: *Workshop on Workflows in Support of Large-Scale Science (WORKS'06)*. Paris, France.
25. Nicolau, S., X. Pennec, L. Soler, and N. Ayache: 2003, 'Evaluation of a New 3D/2D Registration Criterion for Liver Radio-Frequencies Guided by Augmented Reality'. In: *International Symposium on Surgery Simulation and Soft Tissue Modeling (IS4TM'03)*, Vol. 2673 of *LNCS*. Juan-les-Pins, pp. 270–283.
26. Oinn, T., M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. Pocock, A. Wipat, and P. Li: 2004, 'Taverna: A tool for the composition and enactment of bioinformatics workflows'. *Bioinformatics journal* **17**(20), 3045–3054.
27. Pautasso, C., T. Heinis, and G. Alonso: 2006, 'JOpera: Autonomic Service Orchestration'. *IEEE Data Engineering Bulletin* **29**(3).
28. Puget, J., J. Lamarre, M. Sygnet, and et al: 1998, 'Planck HFI'. *Proposal submitted to ESA* **1**(1), 1–166.
29. Taffoni, G., D. Maino, G. deGasperis, and et al: 2005, 'The prototype of a computational Grid for Planck Satellite'. In: *Astronomical Data Analysis Software and Systems (ADASS) XIV*. Pasadena, US, p. 4.
30. Tanaka, Y., H. Nakada, S. Sekiguchi, T. Suzumura, and S. Matsuoka: 2003, 'Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing'. *Journal of Grid Computing (JGC)* **1**(1), 41–51.
31. Taylor, I., I. Wand, M. Shields, and S. Majithia: 2005, 'Distributed computing with Triana on the Grid'. *Concurrency and Computation: Practice & Experience* **17**(1–18).
32. Tweed, T. and S. Miguet: 2003, 'Medical image database on the grid: strategies for data distribution'. In: *HealthGrid'03*. Lyon, France, pp. 152–162.
33. Yu, J. and R. Buyya: 2005, 'A Taxonomy of Workflow Management Systems for Grid Computing'. *Journal of Grid Computing (JGC)* **3**(3-4), 171 – 200.
34. Zaldarriaga, M. and U. Seljak: 2000, 'CMBFAST for Spatially Closed Universes'. *The Astrophysical Journal Supplement Series* **129**(2), 431–434.